Assignment 1 solutions

1a)

$\Theta\left(\frac{\sqrt{\log n}}{n^{1.5}}\right)$	$\frac{\sqrt{n\log n+1}}{n^2+1}$
$\Theta(\log n)$	$\log(n^3+2n)$
$\Theta((\log n)^2)$	$[\log(\frac{n}{2})]^2$
$\Theta(n^{0.63})$	$2^{\log_3 n} = n^{\log_3 2} = n^{0.63}$
$\Theta(n)$	$\log_3 2^n = n\log_3 2$, $2^{\log_2 n+1} = 2\cdot n$
$\Theta(n\log n)$	$\frac{n^2\log n + n}{n+\log n}$
$\Theta(n^{1.5}\sqrt{\log n})$	$\sqrt{n^3\cdot\log n + n}$
$\Theta(2^n)$	$8^{n/3+1} = 8\cdot 2^n$
$\Theta(3^n)$	$(2^{10}+6)(2^n+3^n)$
$\Theta(8^n)$	$2^{3n+1} = 2\cdot 8^n$

b) 1. False. For counter example, consider $f(n)=\sqrt{n}$. Note $\sqrt{n}\le n$, so f is $O(n)$. But $\sqrt{n}\not\geq cn$ for any constant c. $\therefore f$ is not $\Theta(n)$.

2. True. Given that f is $O(n^3)$, then $f\le cn^3$ for some $c>0$, $n\ge k$. $\therefore f\le cn^3\le c\cdot 3^n$ since $n^3\le 3^n$ for all $n\ge 1$ $\therefore f$ is $O(3^n)$

3. False. For counter example, consider $f(n)=n^2$. Since $n^2\le n^2$, f is $O(n^2)$. But $n^2\not\leq cn$ for any c. Therefore f is not $O(n)$.

c) 1. $f = n^2 - \frac{\sqrt{n}}{\log n} + n(\log n)^2$ is $\Theta(n^2)$

Show $f\le c\cdot n^2$

$$f = n^2 - \frac{\sqrt{n}}{\log n} + n(\log n)^2$$
$$\le n^2 + n(n^{1/2})^2 \qquad \log n\le\sqrt{n} \text{ for } n\ge 16$$
$$= 2n^2 \quad \forall n\ge 16$$

Show $f\ge cn^2$

$$f = n^2 - \frac{\sqrt{n}}{\log n} + n(\log n)^2$$
$$\ge n^2 - \frac{n^2}{2} \qquad \sqrt{n}\le n^2 \text{ and } 2\le\log n \ \forall n\ge 4$$
$$= n^2/2 \quad \forall n\ge 4$$

2. $f = \frac{3n^3-n}{2n+\log n} + 2^{10}$ is $\Theta(n^2)$

Show $f\le cn^2$

$$f = \frac{3n^3-n}{2n+\log n} + 2^{10} \le \frac{3n^3}{2n} + 2^{10}n^2$$
$$= 1.5n^2 + 2^{10}n^2$$
$$= (2^{10}+1.5)n^2 \quad \forall n\ge 1$$

Show $f\ge cn^2$

$$f = \frac{3n^3-n}{2n+\log n} + 2^{10}$$
$$\ge \frac{3n^3-n^3}{2n+n} \qquad n^3\ge n, \quad \log n\le n$$
$$= n^2, \ \forall n\ge 1$$

3. $f = 2^n \cdot n + n^5 \log n - 1.5^n$ is $\Theta(2^n \cdot n)$

Show $f \le c \cdot 2^n \cdot n$

$f = 2^n \cdot n + n^5 \log n - 1.5^n$

$\le 2^n \cdot n + 2^n \cdot n$

$= 2 \cdot 2^n \cdot n \quad \forall n \ge 32$

$n^4 \log n \le 2^n$

$\forall n \ge 32$

(tighter value possible)

	$n^4 \log n$	2^n
n=2	16	4
4	512	16
8	12 288	256
16	⋮	⋮
32		

for $n \ge 32$, $2^n \ge n^4 \log n$

Show $f \ge c \cdot 2^n n$

$f = 2^n n + n^5 \log n - 1.5^n$

$\ge 2^n n - 1.5^n$

$\ge 2^n n - \frac{2^n \cdot n}{2}$

$= \frac{1}{2}(2^n \cdot n)$

$\forall n \ge 2$

$2^n \cdot n \ge 2(1.5)^n$

$\forall n \ge 2$

	$2(1.5)^n$	$2^n \cdot n$
n=1	3	2
n=2	4.5	8
n=3	6.75	24

4. $f = \sqrt{n^3+1} + n^2\sqrt{n+1}$ is $\Theta(n^{2.5})$

Show $f \le c \cdot n^{2.5}$

$f = \sqrt{n^3+1} + n^2\sqrt{n+1}$

$\le \sqrt{n^3+n^3} + n^2\sqrt{n+n}$

$\le 2n + n^2(2n^{1/2})$

$\le 2n^{3/2} + 2n^{5/2} = 4n^{5/2}$

$\forall n \ge 1$

Show $f \ge cn^{2.5}$

$f = \sqrt{n^3+1} + n^2\sqrt{n+1}$

$\ge n^2\sqrt{n+1} \ge n^{5/2} \quad \forall n \ge 1$

Q2. SimpleSort(A, s, f)

```
n = f - s + 1
Initialize array L[1...n].
last = -INF
lastindex = 0
end = 0
while lastindex ≠ n
    for i = s to f
        if A[i] > last
            lastindex ++
            last = A[i]
            L[lastindex] = A[i].
            A[i] = -INF        // excludes element A[i]
    Merge(L, 1, end, lastindex)    // note that this returns when end = 0
    end = lastindex
    last = - INF
```

Find worst-case # comparisons:

Consider reversely-sorted input:

iteration 1) $A = 10\ 9\ 8 \ldots$ 1 comp.

2) $A = \cancel{10}\ 9\ 8\ 7\ 6\ 5 \ldots 1 \rightarrow L = [10\ 9 \longrightarrow]$ TOTAL: 2.

comp. compare.

$L = [9\ 10 \longrightarrow]$

3) $A = [\cancel{10}\ \cancel{9}\ \underbrace{8\ 7}_{\text{comp.}} \ldots] \rightarrow L = [\underbrace{9\ 10\ 8}_{\text{comp.}}. -]$ TOTAL: 2

4) $A = [\cancel{10}\ \cancel{9}\ \cancel{8}\ \underbrace{7\ 6}_{\text{comp.}} \ldots]$, $L = [8\ \underbrace{9\ 10\ 7}_{\text{comp.}} -]$ TOTAL: 2.

⋮

∴ Total # comp. for reversely sorted data is $2(n-1)+1$.

There is no guarantee that this is the worst case scenario!
Other input types would need to be evaluated to determine if more comparisons are possible.

* Note: full marks given for analysis of reversely sorted data, even though this might not be the worst case.

Best-case # of comp: (recall we only count comparisons between elements of the input)

Consider elements sorted in increasing order: 1, 2, 3, ... 10.

lastindex = 0:
while loop:
for i = 1 to n
i = 1 last = -INF, element 1 added to L, L = [1 ...]
i = 2 last = 1. Compare 2 to 1. L = [1 2 ...]
i = 3 last = 2. Compare 3 to 2. L = [1 2 3 ...]
⋮
i = n last = n-1. Compare n to n-1. L = [1 2 ... n].
lastindex = n.

$\}$ n-1 comp.

Merge(L, 1, 0, n) } causes no comparisons.
exit while loop since lastindex = n.

Total comparisons: $n-1$ Note that this <u>must</u> represent the best case scenario, since min # of comparisons in any sorting alg is $n-1$.

Algorithm: See "strand sort" on wikipedia. The alg. description is identical.

Runtime:
Best: $O(n)$
Worst: $O(n^2)$
Avg: $O(n^2)$.

b) Execution:

$A = [3, 4, 5, 1, 6, 2]$ $s = 1$ $f = 6$

$e = 1$

first while loop: $e = 3$

2nd while loop:

$e = 3$: $e2 = 4 \rightarrow e2 = 5$

merge$(A, 1, 3, 5) \rightarrow A = [1\ 3\ 4\ 5\ 6\ 2]$

$e = 5$

$e = 5$: $e2 = 6$.

merge$(A, 1, 5, 6) \rightarrow A = [1\ 2\ 3\ 4\ 5\ 6]$

$e = 6$

Exit while loop.

Correctness:

The algorithm is correct because:

- After the first while loop, the section of the array between s and e is sorted (inc.)
- The 2nd while loop executes until $e = f$.
  - At the start of the inner while loop, $A[s \ldots e]$ is sorted
  - After each internal while loop, the section of $A[e+1, \ldots e2]$ is also sorted
  - Merge$(A, s, e, e2)$ results in $A[s, \ldots e2]$ being sorted
  - $e = e2$.

$\therefore$ the algorithm is merging increasing substrings of the array.

Best-case Runtime: Assume $A[1 \ldots n]$ is sorted in increasing order.

After the first while loop, $e = f$. $\therefore$ the 2nd while loop is not executed.

$T(n) = a + \underbrace{(n-1)}_{\#\text{ of iterations of 1st while loop}} \cdot b\}$ runtime of each iteration of while loop.

$= a + bn - b$.

$\leq a + bn \leq an + bn = (a+b)n$ $\therefore T(n)$ is $O(n)$.

Runtime on decreasing array: 10, 9, 8, 7, ...

* Recall merge() runs in time $\leq d \cdot n$ for n elements.

First while loop: exits after 1 iteration since $A[e] > A[e+1]$.

Second while loop: $e = 1$, $e2 = 2$ inner while loop exists. Runtime: c

merge$(A, 1, 1, 2)$: Runtime: $2d$

$e = 2$ $e2 = 3$ inner while loop exits! Runtime: c

merge$(A, 1, 2, 3)$ Runtime: $\leq 3d$

⋮

$e = n-1$ $e2 = n$ while loop exits. Runtime: c

merge $(A, 1, n-1, n)$ Runtime: $\leq n \cdot d$.

$e = n$, exit while loop.

Total Runtime: $c(n-1) + d(2+3+\dots+n)$

$$= c(n-1) + d\left(\frac{n(n+1)}{2} - 1\right)$$

$$\leq cn^2 + d\frac{(n+1)^2}{2} \leq cn^2 + 2dn^2 \leq (c+2d)n^2$$

$\therefore T(n)$ is $O(n^2)$

Q3 a) An in place version of Merge Sort exits, however the runtime is $O(n(\log n)^2)$.

b) MergeThree(A, q1, q2, f) // Assumes s → q1 sorted, q1+1 → q2 sorted, q2+1 → f sorted.

Initialise L[ ], M[ ], R[ ], of size f-s.

O(n): total n elements copied: dn

Copy A[s .. q1] to L[ ]
Copy A[q1+1 .. q2] to M[ ]
Copy A[q2+1 .. f] to R[ ].
L[q1+1] = ∞   M[q2+1] = ∞   R[f+1] = ∞
i = 1, j = 1, k = 1
for m = s to f.

executed n times. constant c

    if L[i] ≤ R[k] and L[i] ≤ M[j]
        A[m] = L[i]
        i++
    else if R[k] ≤ L[i] and R[k] ≤ M[j]
        A[m] = R[k]
        k++
    else A[m] = M[j]
        j++

Total Runtime of MergeThree: $\leq dn + cn \therefore O(n)$

MergeSortThree(A, s, f)
    if f - s ≥ 2 // at least 3 elements.
        $q1 = \left\lfloor \frac{2s+f}{3} \right\rfloor$ $q2 = \left\lfloor \frac{s+2f}{3} \right\rfloor$.

        MergeSortThree(A, s, q1)
        MergeSortThree(A, q1+1, q2)
        MergeSortThree(A, q2+1, f)
        MergeThree(A, q1, q2, f)
    else if f = s+1 // two elements
        if A[s] > A[s+1]
            swap A[s], A[s+1].

(c) $T(n) = \underbrace{3T(n/3)}_{\text{3 rec. calls to input } n/3} + \underbrace{cn}_{\text{Runtime of mergeThree.}}$

Master method: $K = \log_3 3 = 1$ $\quad n^K = n \quad f(n) = cn$

Since both are $\Theta(n)$, runtime is $\Theta(n\log n)$.

(d) $A = [5\ 4\ 3\ 2\ 1]$ $\quad s = 1 \quad f = 5$

MP(A, 1, 5)

A = [5 4 3 2 1]

q1 = 2 $\quad$ q2 = 3

MergeSort(A, 2, 3)

A = [5 3 4 2 1]

MP(A, 2, 5)

A = [5 3 4 2 1]

q1 = 3 $\quad$ q2 = 4

MergeSort(A, 3, 4)

A = [5 3 2 4 1]

MP(A, 3, 5)

MergeSort(A, 3, 5)

A = [5 3 1 2 4]

MP(A, 2, 4)

MergeSort(A, 2, 4)

A = [5 1 2 3 4].

MP(A, 1, 3)

MergeSort(A, 1, 3)

A = [1 2 5 3 4]

Final version of A: [1 2 5 3 4].

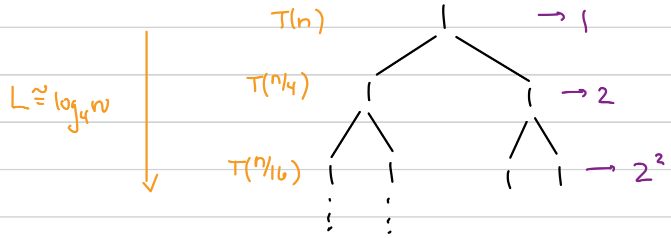
Runtime Recurrence: $T(n) = \underbrace{c\left(\frac{n}{3}\log\left(\frac{n}{3}\right)\right)}_{\text{MergeSort on } n/3} + \underbrace{T(2n/3)}_{\text{Rec. call from } q1 \to f} + \underbrace{T(2n/3)}_{\text{Rec. call } s \to q2.}$

$$T(n) = cn\log n + 2T(2n/3).$$

Master method: $K = \log_{3/2} 2 \approx 1.71$ $\quad f(n) = cn\log n$

$\therefore T(n)$ is $\Theta(n^{1.71})$.

Q4 a) 1. $T(n) = 2T(n/4) + 1$



Total Runtime:

$$\sum_{K=0}^{L} 2^K$$
$$= 2^{L+1} - 1$$
$$= 2^{\log_4 n + 1} - 1$$
$$= 2n^{\log_4 2} - 1$$
$$= 2n^{0.5} - 1$$

$\therefore T(n)$ is $\Theta(\sqrt{n})$

2. $T(n) = 4T(n/4) + n$

$L \cong \log_4 n$

Runtime:

$T(n)$: $n \longrightarrow n$

$T(n/4)$: $n/4 \quad n/4 \quad n/4 \quad n/4 \longrightarrow 4(n/4) = n$

$T(n/16)$: $\frac{n}{16} \cdots\cdots n/16 \longrightarrow 16(\frac{n}{16}) = n$

$\vdots$

Total: $\sum_{K=0}^{L} n$

$\cong n \cdot (\log_4 n)$

$\therefore T(n)$ is $\Theta(n \log n)$

3. $T(n) = T(n/2) + \log n$

$L \cong \log_2 n$

$\log n$

$\log(n/2)$

$\log(n/4)$

$\vdots$

Runtime: $T(n) = \sum_{K=0}^{L} \log(\frac{n}{2^K})$

$\leq \sum_{K=0}^{L} \log(n)$

$= (\log n)(\log_2 n)$

$\therefore T(n)$ is $O((\log n)^2)$.

4. $T(n) = 9T(n/3) + n^2$

$L \cong \log_3 n$

Runtime:

$T(n)$: $n^2 \longrightarrow n^2$

$T(n/3)$: $(\frac{n}{3})^2 \cdots\cdots (\frac{n}{3})^2 \longrightarrow 9(\frac{n}{3})^2 = n^2$

$T(n/9)$: $(\frac{n}{3^2})^2 \cdots\cdots (\frac{n}{3^2})^2 \longrightarrow 9^2(\frac{n^2}{9^2}) = n^2$

Total: $\sum_{K=0}^{L} n^2$

$= n^2(\log n)$

$\therefore T(n)$ is $\Theta(n^2 \log n)$

b) 1. $T(n) = 2T(n/2) + n^2 + n$ $\quad K = \log_2 2 = 1 \quad f(n) = n^2 \quad \therefore T(n)$ is $\Theta(n^2)$

2. $T(n) = 15T(n/4) + n^2 \log n$, $\quad K = \log_4 15 \cong 1.95 \quad f(n) = n^2 \log n \quad \therefore T(n)$ is $\Theta(n^2 \log n)$

3. $T(n) = 17T(n/4) + n^2 + \log n$, $\quad K = \log_4 17 \cong 2.04$, $\quad f(n) = \Theta(n^2) \quad \therefore T(n)$ is $\Theta(n^{2.04})$

4. $T(n) = 16T(n/4) + n^2 \log n + n^3$, $\quad K = \log_4 16 = 2$, $\quad f(n) = \Theta(n^3)$. $\quad \therefore T(n)$ is $\Theta(n^3)$